IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#10
4-15-04

Applicants:	Tinku Acharya, et al.	§	Group Art Unit:	2623
Serial No.:	09/722,982	§	Examiner:	Mehrdad Dastouri
Filed:	November 27, 2000	§		
For:	Computing the Euler Number of a Binary Image	§	Atty. Dkt. No.:	ITL.0493US (P10273)
Customer No.:	21906	§	Confirmation No.:	5885

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

Applicants respectfully appeal from the final rejection mailed January 29, 2004.

I. REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF THE CLAIMS

Claims 1-3, 7, 8, 10-14 have been finally rejected and are the subject of this appeal.

IV. STATUS OF AMENDMENTS

All amendments are believed to have been entered except the one filed on February 5, 2004.

V. SUMMARY OF THE INVENTION

In accordance with the embodiments described herein, an Euler number representation of a binary image is calculated. The calculations may be performed using discrete logic components to analyze the binary image. Specification, at page 3, line 14 through line 16.

In Figure 1, a system 300 includes logic for computing an Euler number to represent a binary image, according to principles described below. In the following, a binary image 16 is analyzed by the system 300 and, based upon the analysis, an Euler number 90 is computed. Specification, at page 3, line 17 through line 20.

The system 300 includes circuitry for calculating the Euler number 90 of the binary image. According to the embodiments described below, a portion of the binary image 16 is analyzed such that Euler number 90 for the portion, or partial binary image, may be computed. Additional portions of the binary image 16 are likewise analyzed, in an iterative manner, until an Euler number 90 of the entire binary image 16 may be obtained. Specification, at page 3, line 21 through line 26.

The Euler number 90 may be very useful in characterizing a binary, or two-tone, image. Such characterization, for example, may be useful in performing database retrieval of binary images. Further, the principles embodied herein may be applied to gray-tone or color images which are divided into two-tone portions. Before describing the features of the system 300 further, an explanation of Euler numbers, as well as some properties of Euler numbers, follows. Specification, at page 4, line 1 through line 7.

In Figure 2, according to one embodiment, a pixel matrix 20 of size N x M, in which N=14 and M=10, is shown. The pixel matrix 20 includes N x M pixels 12 in which each pixel 12 may have a value of either a "1," for object pixels, or a "0," for background pixels. Specification, at page 4, line 8 through line 11.

In arriving at the Euler number 90, an analysis of the binary image 16 within the pixel matrix 20 is made. The Euler number is the difference between the number of connected components, or objects, and the number of holes in the binary image 16. A simple, geometric binary image 16 is depicted in Figure 3. Specification, at page 4, line 12 through line 16.

To determine the number of connected components and holes in the binary image 16, a pixel 12 is analyzed in terms of adjacent pixels, or its neighbors, of variable granularity. For a pixel 12 at location (x,y) in the pixel matrix 20, a 4-neighborhood 14 includes the set of pixels at locations [(x-1,y), (x+1,y), (x,y-1), and (x,y+1)]. For example, in Figure 2, a 4-neighborhood 14 of pixel locations [(2,2), (4,2), (3,1), and (3,3)] surrounds the pixel 12 at location (3,2). Specification, at page 4, line 17 through line 23.

Similarly, an 8-neighborhood 18 of pixel locations [(8,5),(8,6),(8,7),(9,5), (9,7), (10,5), (10,6), (10,7)] is shown in Figure 2, for the pixel 12 at location (9,6). As expected, the 8-neighborhood 18 includes eight pixels 12 surrounding the pixel 12 that is analyzed. The 8-neighborhood 18 of a pixel 12 at location (x,y) includes the set of pixels at locations [(x-1,y-1), (x-1,y), (x-1,y+1), (x,y-1), (x,y+1), (x+1,y-1), (x+1,y), and (x+1,y+1)]. Neighborhoods of different sizes may similarly be defined. Specification, at page 4, line 24 through page 5, line 2.

In Figure 3, the binary image 16, a diamond-like geometric pattern with a hole in its center, occupies the pixel matrix 20. In the binary image 16, a connected component is a set of object pixels, e.g., pixels with a value of "1," such that any object pixel in the set is in the neighborhood of at least one object pixel of the same set. Specification, at page 5, line 3 through line 7.

Assume that the shaded pixels represent a pixel value of "1." All of the shaded pixels shown in the binary image 16 of Figure 3 are connected components for a 8-neighborhood 18. However, for a 4-neighborhood 14, the shaded pixels 12 at (4,2), (10,2), (4,8), and (10,8) would not be considered connected components to the set comprising the remaining shaded pixels 12. Thus, the size of the neighborhood may affect which pixels 12 are connected to other pixels 12. Specification, at page 5, line 8 through line 14.

Another feature of the binary image 16 used to calculate the Euler number is a hole. A hole is set of background pixels, e.g., pixels of value "0" such that any background pixel in the set is in the neighborhood of at least one background pixel of the same set. Further, the entire set of background pixels is enclosed by a connected component. Specification, at page 5, line 15 through line 19.

Thus, in Figure 3, the five non-shaded pixels 12 at the center of the binary image 16, or pixels at positions (6,5), (7,4), (7,5), (7,6), and (8,5), together make up the lone hole of the binary image 16. Because a hole is enclosed by a connected component, pixels 12 at position (5,2) and (10,7) are not considered hole pixels. It may be noted that, for an 8-neighborhood object, a hole should have a 4-neighborhood relation, and vice-versa. Specification, at page 5, line 20 through line 25.

Still another feature used to calculate the Euler number, according to one embodiment, is a run. A run, R(i), of the i^{th} column (or row) of the pixel matrix 20 is defined to be a maximal sequence of consecutive 1's in the i^{th} column (or row). Specification, at page 5, line 26 through page 6, line 2.

If the binary image 16, known as I, consists of a single row or a single column i, the Euler number of the image 16 is the same as the number of runs, R(i), of the image 16, as shown by the following formula:

$$\text{Property A: } E(I) = R(i)$$

Specification, at page 6, line 3 through line 6.

For example, for the 0th row of the pixel matrix 20 of Figure 3, the run R(0) is zero, since there are no consecutive 1's in the row. For the first row, the number of runs, R(1), equals one. In the second row, the number of runs, R(2), is three, as there are three maximal sequences of 1s: a single '1,' at position (4,2), a sequence of three consecutive 1's, (from 6,2) through (8,2), and a single '1,' at position (10,2). Specification, at page 6, line 7 through line 12.

Thus, from Property A, the Euler number, E(I), of image I may be determined, where I includes a single row (or column). Additionally, the Euler number satisfies the additive set property. Given two images, I_1 and I_2, the union ($\cup$) of the two images is defined as a simple

juxtaposition of I_1 and I_2, either vertically or horizontally, without any overlap. Specification, at page 6, line 13 through line 17.

For example, an image, I_1, in Figure 4A includes pixel rows 1 through k. Image I_2, in Figure 4B, includes rows 1 through m. The union of these images, $I_1 \cup I_2$, is illustrated in Figure 4C. Notice that the resulting image $I_1 \cup I_2$ includes k+m pixel rows. Specification, at page 6, line 18 through line 21.

In contrast, the intersection ($\cap$) of I_1 and I_2 is the image formed by the last row (or column) of I_1, and the first row (or column) of I_2, if the images I_1 and I_2 are joined horizontally (or vertically). The intersection image $I_1 \cap I_2$ is illustrated in Figure 4D. Note that the intersection image is always two pixel rows (or columns) wide. Specification, at page 6, line 22 through line 26.

Suppose binary images I_1 and I_2 are joined horizontally, i.e., I_1 lies entirely above I_2, as shown in Figure 4C. The last row of I_1 lies above the first row of I_2. Two runs appearing in two adjacent rows each are said to be neighboring if at least one pixel 12 of a run is in the neighborhood of a pixel 12 of the other run. Neighboring runs are defined according to the type of neighborhood, whether an 8-neighborhood, a 4-neighborhood, or a neighborhood of some other size. Specification, at page 6, line 27 through page 7, line 4.

Restating the formula, the Euler number of the binary image 16 is the difference between the sum of the number of runs for all rows (or columns) and the sum of the number of neighboring runs between all consecutive pairs of rows (or columns) of the pixel matrix 20. Specification, at page 8, line 2 through line 5.

Looking back to Figure 1, according to one embodiment, the system 300 includes a run processor 70, for calculating the number of runs for the rows (or columns) of the binary image 16. The run processor 70 is described in more detail in Figure 7, below. Specification, at page 8, line 6 through line 9.

The system 300 further includes a neighboring run processors 80, in one embodiment, for calculating the number of neighboring runs for consecutive rows (or columns) of the binary

image 16. The neighboring run processor 80 is described in more detail in Figure 8, below. Specification, at page 8, line 10 through line 13.

A math processor 66 computes the sum of the number of runs and the sum of the number of neighboring runs, according to one embodiment. The math processor 66 additionally may subtract the two sums, as specified above, in computing $E(I_N)$. The run processor 70, neighboring run processor 80, and math processor 66 may be implemented using discrete logic, specialized processing hardware such as digital signal processors (DSPs), or software, to name a few possible implementations of the system 300. Specification, at page 8, line 14 through line 20.

Using the formula for calculating the Euler number, $E(I_N)$, above, the Euler number of the binary image 16 may be iteratively computed. The Euler number of the first row (or column), comprising a first or partial image, is calculated. The next row (or column) is then added to the partial image and then the Euler number of the union image is calculated. Thereafter, successive rows (or columns) are added together and, accordingly, the Euler number of each partial image is calculated. The process is repeated until all rows (or columns) are exhausted. Specification, at page 8, line 21 through line 28.

In Figure 5, a flow diagram illustrates the calculation of the Euler number for the binary image 16, according to one embodiment. The number of runs, R(1), in the first row (or column) of the binary image 60 of the pixel matrix 20 is computed (block 102). A first Euler number, E(I), for the partial image occupying only the first row (or column) is calculated, using property A (block 104). The variable, i, denoting the row (or column) of the pixel matrix 20, is set for subsequent analysis of the binary image 16 (block 106). Specification, at page 9, line 1 through line 7.

Succeeding operations are performed iteratively for all i rows (or columns) of the binary image 16. The number of runs, R(i), in the i^{th} row (or column) is calculated (block 108). Likewise, the number of neighboring runs, O_i, between the $(i-1)^{th}$ and the i^{th} rows (or columns) is calculated (block 110). Specification, at page 9, line 8 through line 11.

Once the number of runs, R(i) and the number of neighboring runs, O_i, is known, a new Euler number, E(I), of a partial image consisting of the first and subsequent rows (or columns) may be calculated, according to the formula derived above (block 112). This process is repeated iteratively until all the rows (or columns) of the pixel matrix 20 have been analyzed. Specification, at page 9, line 12 through line 16.

In a second embodiment, multiple run and neighboring run number calculations are performed in parallel, as shown in Figure 6. Although the binary image 16 is subdivided into partial images according to rows, the binary image 16 may instead be divided into partial images comprising columns. Specification, at page 9, line 17 through line 20.

While the number of runs in the 1st row is calculated, the numbers of runs in the 2nd row may be calculated, the number of runs in the 3rd row may be calculated, and so on up to the Nth row, where N parallel processors are available (block 204). Specification, at page 9, line 21 through line 24.

Likewise, the number of neighboring runs between the 1st and the 2nd rows may be calculated, while, simultaneously with the operations of block 204, the number of neighboring runs between the 2nd and 3rd rows, between the (N-1)th and Nth rows, and so on are calculated, where (N-1) neighboring run processors are available (block 206). Where N-1 processors are not available, a smaller number of parallel operations may be performed. The variable, i (shown in block 202), could be incremented based upon the number of parallel operations being performed. The operations of blocks 204 and 206 may be repeated, as needed, until both the number of runs and the number of neighboring runs for all rows have been calculated. Specification, at page 9, line 25 through page 10, line 6.

In one embodiment, the runs for all the rows are added together (block 208). Likewise, the number of neighboring runs for all pair of consecutive rows are added together (block 210). The Euler number 90 for the complete binary image 16 is calculated by subtracting the sum of the neighboring runs from the sum of the neighboring runs (block 212). Specification, at page 10, line 7 through line 11.

In Figure 7, a run processor 70, according to one embodiment, includes discrete elements for identifying runs within the binary image 16. The discrete elements include a D flip-flop 34, an AND gate 44, and a modulo counter 42. Specification, at page 10 line 21 through line 23.

To process an N x M image in parallel, according to one embodiment, N copies of the run processor 70 and N-1 copies of the neighboring run processor 80 are used. For example, in Figure 9, a circuit for calculating the Euler number for a 4 x 5 image includes four run processors 70 and three neighboring run processors 80. Specification, at page 12, line 27 through page 13, line 3.

Each neighboring run processor 80 receives three sets of inputs, according to one embodiment. Two input signals 30 correspond to each pixel in adjacent rows (or column) of the pixel matrix 20. Two X outputs 40 and two Q outputs 36 correspond to the neighboring relation of a run for the adjacent rows (or columns). Specification, at page 13, line 4 through line 8.

The system 300 of Figure 1 may be implemented in a processor-based system 400, as in Figure 10. A processor 24 is connected to the system 300 by a bus 26. The system 300 may serve as a coprocessor to the processor 24, to compute the Euler number 90. Specification, at page 13, line 22 through line 25.

VI. ISSUES

A. Is claim 1 anticipated by the Di Zenzo reference?

B. Is claim 2 anticipated by the Di Zenzo reference?

C. Is claim 7 anticipated by the Di Zenzo reference?

VII. GROUPING OF THE CLAIMS

For purposes of this appeal, claim 3, 12-14 may be grouped with claim 1 and claims 8, 10, 11 may be grouped with claim 7. The patentability of each group and claim 2 is discussed below.

VIII. ARGUMENT

A. Is claim 1 anticipated by the Di Zenzo reference?

In the Di Zenzo reference there is no teaching of identifying a representation of a binary image in the pixel matrix wherein the Euler number of the binary image is computed. However, the claim language recites "a representation of a binary image in a pixel matrix" and "computing the Euler number." Therefore, the Di Zenzo reference fails to show that the representation of a binary image is identified in a pixel matrix where Euler number of the binary image is computed. That is, the Di Zenzo reference does not teach a pixel matrix based computing of the Euler number of a binary image.

In the Di Zenzo reference, the binary image is represented by a plurality of 1 x m rectangles depicted in graph C in Figure 1(a) called a graph representation of binary image. Figure 1(a) shows the number of runs which are adjacent, i.e. lie in adjacent columns and touch one another. However, the binary image itself, developed from those runs, is not represented as a pixel matrix of columns or columns and rows. Instead, the binary image is represented by the graphs shown in Figure 1which do not meet any recognizable conception of a matrix.

Therefore, the Applicants respectfully request reversal of the Section 102 rejection of claim 1.

B. Is claim 2 anticipated by the Di Zenzo reference?

The method of claim 2 includes computing the number of runs for a first portion of the pixel matrix. The method further comprises identifying one or more runs of the first portion and counting the number of runs. The system of the Di Zenzo reference cannot identify runs of a particular portion of the pixel matrix when no use of portions a pixel matrix is disclosed. The question of counting the number of runs does not arise absent the use of particular portions of the

pixel matrix. Accordingly, the method of claim 2 is not anticipated by the Di Zenzo teachings. Therefore, reversal of the rejection of claim 2 is respectfully requested.

C. Is claim 7 anticipated by the Di Zenzo reference?

Claim 7 calls for a system comprising a run processor to compute a run number, wherein the run number is the number of runs in a portion of a pixel matrix. The system further comprises a neighboring run processor to compute a neighboring run number, wherein the neighboring run number is the number of neighboring runs between the portion and a second portion of the pixel matrix and the neighboring run processor receives a plurality of signals from the run processor.

Since the Di Zenzo reference does not teach use of portions of a pixel matrix, the run number in the number of runs in a particular portion of pixel matrix as indicated in the claim 7 limitations is not taught by the Di Zenzo reference. Likewise, the second limitation of claim 7 is not taught or suggested by the Di Zenzo reference because particular portions of a pixel matrix, as claimed in claim 7 are not suggested or taught by the Di Zenzo reference. In other words, absent a use a plurality of portions of a pixel matrix, the claim 7 limitations are not anticipated by the Di Zenzo reference. Therefore, reversal of the rejection of claim 7 is respectfully requested.

IX. CONCLUSION

Applicant respectfully requests that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,

Date: April 6, 2004

Sanjeev K. Singh Under 37 C.F.R. § 10.9(b)
Registration No. 28,994
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Suite 100
Houston, TX 77024-1805
713/468-8880 [Phone]
713/468-8883 [Fax]

APPENDIX OF CLAIMS

1. A method comprising:

identifying a representation of a binary image in a pixel matrix, wherein the pixel matrix comprises a plurality of portions;

computing the number of runs for a first portion of the pixel matrix, wherein a run is a maximal sequence of pixels having a predetermined value in the first portion;

computing the number of neighboring runs between the first portion and a second portion of the pixel matrix, wherein a neighboring run is a run in which at least one pixel of the run is in the neighborhood of a run in an adjacent portion; and

computing the Euler number from the number of runs and the number of neighboring runs.

2. The method of claim 1, computing the number of runs for a first portion of the pixel matrix further comprising:

identifying one or more runs of the first portion;

counting the number of runs.

3. The method of claim 2, computing the number of neighboring runs between the first portion and a second portion of the pixel matrix further comprising:

determining a neighborhood size;

identifying a second run in the second portion; and

determining whether at least one pixel of the second run is in the neighborhood of the run.

4. The method of claim 3, computing the Euler number from the number of runs and the number of neighboring runs further comprising:

subtracting the number of neighboring runs between the first portion and the second portion from a sum of the number of runs in the first portion and the second portion to arrive at a result; and

adding the result to an Euler number for a third portion.

5. The method of claim 4, computing the Euler number from the number of runs and the number of neighboring runs further comprising:

subtracting from the result the number of neighboring runs between the third portion and a prior third portion.

6. The method of claim 5, identifying the binary image in a pixel matrix further comprising identifying pixels of the predetermined value.

7. A system comprising:

a run processor to compute a run number, wherein the run number is the number of runs in a portion of a pixel matrix; and

a neighboring run processor to compute a neighboring run number, wherein the neighboring run number is the number of neighboring runs between the portion and a second portion of the pixel matrix and the neighboring run processor receives a plurality of signals from the run processor.

8. The system of claim 7, further comprising:

a second run processor to compute a second run number;

a second neighboring run processor to compute a second neighboring run number; and

a math processor.

9. The system of claim 8, wherein the math processor further comprises:

a run adder to add the run number to the second run number;

a neighboring run adder to add the neighboring run number to the second neighboring run number; and

a subtractor to subtract the run number from the neighboring run number.

10. The system of claim 7, wherein the portions comprise rows of the pixel matrix.

11. The system of claim 7, wherein the portions comprise columns of the pixel matrix.

12. An article comprising a medium storing software for enabling a processor-based system to:

identify a binary image in a pixel matrix, wherein the pixel matrix comprises a plurality of portions;

compute the number of runs for a first portion of the pixel matrix;

compute the number of neighboring runs between the first portion and a second portion of the pixel matrix; and

compute an Euler number from the number of runs and the number of neighboring runs.

13. The article of claim 12, further storing software for enabling a processor-based system to compute the number of runs for a first portion of the pixel matrix by:

identifying one or more runs of the first portion, wherein a run is a maximal sequence of pixels with a predetermined value in the first portion;

counting the number of runs.

14. The article of claim 13, further storing software for enabling a processor-based system to compute the number of neighboring runs between the first portion and a second portion of the pixel matrix by:

determining a neighborhood size;

identifying a second run in the second portion; and

determining whether at least one pixel of the second run is in the neighborhood of the run.

15. The article of claim 14, further storing software for enabling a processor-based system to compute the Euler number from the number of runs and the number of neighboring runs by:

subtracting the number of neighboring runs between the first portion and the second portion from a sum of the number of runs in the first portion and the second portion to arrive at a result; and

adding the result to an Euler number for a third portion.

16. The article of claim 15, further storing software for enabling a processor-based system to compute the Euler number from the number of runs and the number of neighboring runs by:

subtracting from the result the number of neighboring runs between the third portion and a prior third portion.